

日本国特許庁  
JAPAN PATENT OFFICE

*Priority Papers*

J1040 U.S. PT  
10/047958  
01/15/02

別紙添付の書類に記載されている事項は下記の出願書類に記載されて  
いる事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed  
with this Office

出願年月日

Date of Application:

2001年 2月 2日

出願番号

Application Number:

特願2001-027462

出願人

Applicant(s):

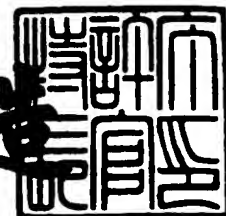
インターナショナル・ビジネス・マシーンス・コーポレーシ  
ョン

CERTIFIED COPY OF  
PRIORITY DOCUMENT

2001年 6月20日

特許庁長官  
Commissioner,  
Japan Patent Office

及川耕造



出証番号 出証特2001-3058446

【書類名】 特許願

【整理番号】 JP9000330

【提出日】 平成13年 2月 2日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 7/00

【発明者】

【住所又は居所】 神奈川県大和市下鶴間 1 6 2 3 番地 1 4 日本アイ・ピー・エム株式会社 東京基礎研究所内

【氏名】 今村 剛

【特許出願人】

【識別番号】 390009531

【氏名又は名称】 インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】

【識別番号】 100086243

【弁理士】

【氏名又は名称】 坂口 博

【代理人】

【識別番号】 100091568

【弁理士】

【氏名又は名称】 市位 嘉宏

【代理人】

【識別番号】 100106699

【弁理士】

【氏名又は名称】 渡部 弘道

【復代理人】

【識別番号】 100112520

【弁理士】

【氏名又は名称】 林 茂則

【選任した復代理人】

【識別番号】 100110607

【弁理士】

【氏名又は名称】 間山 進也

【手数料の表示】

【予納台帳番号】 091156

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9706050

【包括委任状番号】 9704733

【包括委任状番号】 0004480

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 XMLデータの符号化方法、符号化されたXMLデータの復号化方法、XMLデータの符号化システム、符号化されたXMLデータの復号化システム、プログラムおよび記録媒体

【特許請求の範囲】

【請求項1】 XMLデータの文法を定義する文法定義をASN.1抽象構文（型）に変換するステップと、

前記XMLデータを、要素の内容（テキスト）と構造（構造を含む要素名）とに分離するステップと、

前記構造を、前記ASN.1抽象構文（型）に従うASN.1抽象構文（値）に変換するステップと、

前記ASN.1抽象構文（値）をASN.1転送構文に変換するステップと、

前記要素の内容を圧縮するステップと、

圧縮された前記要素の内容と前記ASN.1転送構文とを併合するステップと

を含むXMLデータの符号化方法。

【請求項2】 前記文法定義に属性、処理命令その他前記要素以外の文法定義項目を有し、

前記要素以外の文法定義項目を特別要素として前記要素に含めるように前記文法定義を他の文法定義に変換するステップと、

前記他の文法定義に従うように、前記XMLデータを他のXMLデータに変換するステップと、

をさらに有する請求項1記載のXMLデータの符号化方法。

【請求項3】 前記文法定義はDTDであり、

前記DTDの要素内容には、「,」、「|」、「?」、「\*」または「+」から選択される単一または複数のオペレータが含まれ、または前記オペレータを含まず、

前記ASN.1抽象構文（型）において、

前記「,」オペレータは、sequence型で表現し、

前記「|」オペレータは、`choice`型で表現し、

前記「?」オペレータは、`sequence`型とキーワード「`OPTIONAL`」の組み合わせで表現し、

前記「\*」オペレータは、`sequence-of`型で表現し、

前記「+」オペレータは、サイズが制限された`sequence-of`型で表現し、

前記オペレータが無いときには、`defined`型で表現する、

請求項1または2記載のXMLデータの符号化方法。

【請求項4】 前記文法定義を他の文法定義に変換するステップにおいて、前記文法定義に含まれる属性を、一意に決定できる属性要素として表現し、前記属性の親要素の子要素として扱い、

前記属性の属性値は、「`CDATA`」として前記属性要素の子要素として扱い

前記属性が「`REQUIRED`」属性の場合、前記属性要素を要素で表現し、

前記属性が「`IMPLIED`」属性の場合、または、前記属性にデフォルト値が定義されている場合、前記属性要素を「?」オペレータが適用された要素で表現し、

前記XMLデータを他のXMLデータに変換するステップにおいて、

前記XMLデータの要素に含まれる属性を、一意に決定できる属性要素として表現し、前記属性の親要素の子要素として扱う（ただし前記属性にデフォルト値が定義され、かつ前記属性の属性値が前記デフォルト値と一致する場合を除く）

請求項3記載のXMLデータの符号化方法。

【請求項5】 前記ASN.1抽象構文（値）を前記ASN.1転送構文に変換するステップにおいて、PER規則を用いる請求項1または2記載のXMLデータの符号化方法。

【請求項6】 XMLデータの文法を定義する文法定義をASN.1抽象構文（型）に変換するステップと、

符号化されたXMLデータを、ASN.1転送構文と圧縮された要素の内容（

テキスト) とに分離するステップと、

前記 A S N. 1 転送構文を、前記 A S N. 1 抽象構文 (型) に従う A S N. 1 抽象構文 (値) に変換するステップと、

前記 A S N. 1 抽象構文 (値) を、前記文法定義に従う XML データの構造 (構造を含む要素名) に変換するステップと、

前記圧縮された要素の内容を解凍するステップと、

前記解凍された要素の内容と前記 XML データの構造とを併合するステップと

を含む符号化された XML データの復号化方法。

【請求項 7】 前記文法定義に属性、処理命令その他前記要素以外の文法定義項目を有し、

前記要素以外の文法定義項目を特別要素として前記要素に含めるように前記文法定義を他の文法定義に変換するステップと、

前記文法定義に従うように、前記復号化された XML データを他の XML データに変換するステップと、

をさらに有する請求項 6 記載の符号化された XML データの復号化方法。

【請求項 8】 前記文法定義は D T D であり、

前記 D T D の要素内容には、「, 」、「|」、「?」、「\*」または「+」から選択される単一または複数のオペレータが含まれ、または前記オペレータを含まず、

前記 A S N. 1 抽象構文 (型) において、

前記「, 」オペレータは、sequence 型で表現し、

前記「|」オペレータは、choice 型で表現し、

前記「?」オペレータは、sequence 型とキーワード「OPTIONAL」の組み合わせで表現し、

前記「\*」オペレータは、sequence-of 型で表現し、

前記「+」オペレータは、サイズが制限された sequence-of 型で表現し、

前記オペレータが無いときには、defined 型で表現する、

請求項 6 または 7 記載の符号化された XML データの復号化方法。

【請求項 9】 前記文法定義を他の文法定義に変換するステップにおいて、  
前記文法定義に含まれる属性を、一意に決定できる属性要素として表現し前記  
属性の親要素の子要素として扱い、

前記属性の属性値は、「CDATA」として前記属性要素の子要素として扱い

前記属性が「REQUIRED」属性の場合、前記属性要素を要素で表現し、  
前記属性が「IMPLIED」属性の場合、または、前記属性にデフォルト値  
が定義されている場合、前記属性要素を「？」オペレータが適用された要素で表  
現し、

前記復号化された XML データを他の XML データに変換するステップにおい  
て、

前記復号化された XML データの要素に子要素として含まれる前記属性要素を  
、前記要素の属性またはその属性値に変換する、

請求項 8 記載の符号化された XML データの復号化方法。

【請求項 10】 前記 ASN. 1 転送構文を前記 ASN. 1 抽象構文（値）  
に変換するステップにおいて、PER 規則を用いる請求項 6 または 7 記載の符号  
化された XML データの復号化方法。

【請求項 11】 XML データの文法を定義する文法定義を ASN. 1 抽象  
構文（型）に変換する手段と、

前記 XML データを、要素の内容（テキスト）と構造（構造を含む要素名）と  
に分離する手段と、

前記構造を、前記 ASN. 1 抽象構文（型）に従う ASN. 1 抽象構文（値）  
に変換する手段と、

前記 ASN. 1 抽象構文（値）を ASN. 1 転送構文に変換する手段と、

前記要素の内容を圧縮する手段と、

圧縮された前記要素の内容と前記 ASN. 1 転送構文とを併合する手段と、

を含む XML データの符号化システム。

【請求項 12】 前記文法定義に属性、処理命令その他前記要素以外の文法

定義項目を有し、

前記要素以外の文法定義項目を特別要素として前記要素に含めるように前記文法定義を他の文法定義に変換する手段と、

前記他の文法定義に従うように、前記XMLデータを他のXMLデータに変換する手段と、

をさらに有する請求項11記載のXMLデータの符号化システム。

【請求項13】 前記文法定義はDTDであり、

前記DTDの要素内容には、「,」、「|」、「?」、「\*」または「+」から選択される単一または複数のオペレータが含まれ、または前記オペレータを含まず、

前記ASN.1抽象構文(型)において、

前記「,」オペレータは、sequence型で表現し、

前記「|」オペレータは、choice型で表現し、

前記「?」オペレータは、sequence型とキーワード「OPTIONAL」の組み合わせで表現し、

前記「\*」オペレータは、sequence-of型で表現し、

前記「+」オペレータは、サイズが制限されたsequence-of型で表現し、

前記オペレータが無いときには、defined型で表現する、

請求項11または12記載のXMLデータの符号化システム。

【請求項14】 前記文法定義を他の文法定義に変換する手段において、

前記文法定義に含まれる属性を、一意に決定できる属性要素として表現し、前記属性の親要素の子要素として扱い、

前記属性の属性値は、「CDATA」として前記属性要素の子要素として扱い

前記属性が「REQUIRED」属性の場合、前記属性要素を要素で表現し、

前記属性が「IMPLIED」属性の場合、または、前記属性にデフォルト値が定義されている場合、前記属性要素を「?」オペレータが適用された要素で表現し、



前記XMLデータを他のXMLデータに変換する手段において、

前記XMLデータの要素に含まれる属性を、一意に決定できる属性要素として表現し、前記属性の親要素の子要素として扱う（ただし前記属性にデフォルト値が定義され、かつ前記属性の属性値が前記デフォルト値と一致する場合を除く）

請求項 1 3 記載のXMLデータの符号化システム。

【請求項 1 5】 前記ASN. 1 抽象構文（値）を前記ASN. 1 転送構文に変換する手段において、PER規則を用いる請求項 1 1 または 1 2 記載のXMLデータの符号化システム。

【請求項 1 6】 XMLデータの文法を定義する文法定義をASN. 1 抽象構文（型）に変換する手段と、

符号化されたXMLデータを、ASN. 1 転送構文と圧縮された要素の内容（テキスト）とに分離する手段と、

前記ASN. 1 転送構文を、前記ASN. 1 抽象構文（型）に従うASN. 1 抽象構文（値）に変換する手段と、

前記ASN. 1 抽象構文（値）を、前記文法定義に従うXMLデータの構造（構造を含む要素名）に変換する手段と、

前記圧縮された要素の内容を解凍する手段と、

前記解凍された要素の内容と前記XMLデータの構造とを併合する手段と、

を含む符号化されたXMLデータの復号化システム。

【請求項 1 7】 前記文法定義に属性、処理命令その他前記要素以外の文法定義項目を有し、

前記要素以外の文法定義項目を特別要素として前記要素に含めるように前記文法定義を他の文法定義に変換する手段と、

前記文法定義に従うように、前記復号化されたXMLデータを他のXMLデータに変換する手段と、

をさらに有する請求項 1 6 記載の符号化されたXMLデータの復号化システム

【請求項 1 8】 前記文法定義はDTDであり、

前記 DTD の要素内容には、「,」、「|」、「?」、「\*」または「+」から選択される単一または複数のオペレータが含まれ、または前記オペレータを含まず、

前記 ASN. 1 抽象構文 (型) において、

前記「,」オペレータは、sequence 型で表現し、

前記「|」オペレータは、choice 型で表現し、

前記「?」オペレータは、sequence 型とキーワード「OPTIONAL」の組み合わせで表現し、

前記「\*」オペレータは、sequence-of 型で表現し、

前記「+」オペレータは、サイズが制限された sequence-of 型で表現し、

前記オペレータが無いときには、defined 型で表現する、

請求項 16 または 17 記載の符号化された XML データの復号化システム。

【請求項 19】 前記文法定義を他の文法定義に変換する手段において、

前記文法定義に含まれる属性を、一意に決定できる属性要素として表現し、前記属性の親要素の子要素として扱い、

前記属性の属性値は、「CDATA」として前記属性要素の子要素として扱い

、  
前記属性が「REQUIRED」属性の場合、前記属性要素を要素で表現し、

前記属性が「IMPLIED」属性の場合、または、前記属性にデフォルト値が定義されている場合、前記属性要素を「?」オペレータが適用された要素で表現し、

前記復号化された XML データを他の XML データに変換する手段において、

前記復号化された XML データの要素に子要素として含まれる前記属性要素を、前記要素の属性またはその属性値に変換する、

請求項 18 記載の符号化された XML データの復号化システム。

【請求項 20】 前記 ASN. 1 転送構文を前記 ASN. 1 抽象構文 (値) に変換する手段において、PER 規則を用いる請求項 16 または 17 記載の符号化された XML データの復号化システム。

【請求項 2 1】 コンピュータが実行可能なプログラムであって、コンピュータに、

XMLデータの文法を定義する文法定義をASN.1抽象構文(型)に変換する機能と、

前記XMLデータを、要素の内容(テキスト)と構造(構造を含む要素名)とに分離する機能と、

前記構造を、前記ASN.1抽象構文(型)に従うASN.1抽象構文(値)に変換する機能と、

前記ASN.1抽象構文(値)をASN.1転送構文に変換する機能と、

前記要素の内容を圧縮する機能と、

圧縮された前記要素の内容と前記ASN.1転送構文とを併合する機能と、  
を実現させるためのプログラム。

【請求項 2 2】 コンピュータが実行可能なプログラムであって、コンピュータに、

XMLデータの文法を定義する文法定義をASN.1抽象構文(型)に変換する機能と、

符号化されたXMLデータを、ASN.1転送構文と圧縮された要素の内容(テキスト)とに分離する機能と、

前記ASN.1転送構文を、前記ASN.1抽象構文(型)に従うASN.1抽象構文(値)に変換する機能と、

前記ASN.1抽象構文(値)を、前記文法定義に従うXMLデータの構造(構造を含む要素名)に変換する機能と、

前記圧縮された要素の内容を解凍する機能と、

前記解凍された要素の内容と前記XMLデータの構造とを併合する機能と、  
を実現させるためのプログラム。

【請求項 2 3】 コンピュータ読取可能な記録媒体であって、

コンピュータに、XMLデータの文法を定義する文法定義をASN.1抽象構文(型)に変換する機能と、前記XMLデータを、要素の内容(テキスト)と構造(構造を含む要素名)とに分離する機能と、前記構造を、前記ASN.1抽象

構文（型）に従うASN.1抽象構文（値）に変換する機能と、前記ASN.1抽象構文（値）をASN.1転送構文に変換する機能と、前記要素の内容を圧縮する機能と、圧縮された前記要素の内容と前記ASN.1転送構文とを併合する機能と、を実現させるための第1プログラム、または、

コンピュータに、XMLデータの文法を定義する文法定義をASN.1抽象構文（型）に変換する機能と、符号化されたXMLデータを、ASN.1転送構文と圧縮された要素の内容（テキスト）とに分離する機能と、前記ASN.1転送構文を、前記ASN.1抽象構文（型）に従うASN.1抽象構文（値）に変換する機能と、前記ASN.1抽象構文（値）を、前記文法定義に従うXMLデータの構造（構造を含む要素名）に変換する機能と、前記圧縮された要素の内容を解凍する機能と、前記解凍された要素の内容と前記XMLデータの構造とを併合する機能と、を実現させるための第2プログラム、

の何れかのプログラムが記録された記録媒体。

#### 【発明の詳細な説明】

##### 【0001】

#### 【発明の属する技術分野】

本発明は、データの符号化および復号化技術に関し、特にXML (extensible markup language)データの符号化における圧縮率の向上に適用して有効な技術に関する。

##### 【0002】

#### 【従来の技術】

近年、インターネット上でのデータ表現の手段としてXMLが注目されている。XMLは拡張可能なメタ言語であり、ユーザが独自に文法を規定することができる。また、各要素に論理的な意味を持たせることが可能であり、HTML (hypertext markup language)に比較してデータ処理が大幅に容易になる。このため、電子商取引等インターネットで交換される構造化文書の標準的な表現方式として期待されている。なお、XMLについては、「W3C. Extensible Markup Language (XML) 1.0, 1998. <http://www.w3.org/TR/REC-xml>」に詳細が記述されている。

## 【0003】

XMLデータは文字で記述されるため可読性が高い利点がある。その反面、冗長性も高いという難点がある。すなわち要素の意味は主に開始タグ内に記述され、その意味内容は文字で記述されている内容を参照すれば容易に把握できる。しかし、全てが文字で記述されるため、全体の文字数が増加し、XMLデータ全体の情報量（文字数）が多くなる。文字数が多ければ、たとえばストレージに格納したり、あるいはネットワークで転送する際に、記録容量あるいは転送量が多くなり物理的、時間的コストの上昇を来す。そこで、XMLデータを短い符号に符号化（または圧縮）できれば都合がよい。

## 【0004】

データの圧縮法には各種の手法が知られている。たとえば、ランレングス符号、Huffman符号、算術符号、LZ77等である。これら符号化の手法については、たとえば、「Huffman, D.A. "A method for the construction of minimum-redundancy codes" Proc. of the IRE September, 1952」、 「Mark Nelson and Jean Loup Gailly "The Data Compression Book", Second Edition. M&T Books 1996」、 「Jacob Ziv and Abraham Lempel. "A universal algorithm for sequential data compression" IEEE Transactions on Information Theory May, 1977」に詳しい。

## 【0005】

しかしながら、これら圧縮手法はXMLに特化されたものではなく、XMLデータに適用した場合に必ずしも圧縮効率が低いものではない。XMLデータに特化した圧縮手法には、たとえば「D. Suciú and H. Liefke. XMill: an Efficient Compressor for XML Data, 1999. <http://www.research.att.com/sw/tools/xmill/>」に記載のXMill、「XML Solutions Corp. XMLZip, 1999. <http://www.xmls.com/products/xmlzip/xmlzip.html>」に記載のXMLZip、あるいは「井川甲作著、東京工業大学工学部情報工学科卒業論文「DTDを用いたXML文書圧縮アルゴリズムに関する研究」、平成12年2月」に記載のXCompなどがある。

## 【0006】

XM i l l は、XMLデータから要素ごとのコンテンツ（テキスト）部分を抽出する。この抽出された部分をコンテナと呼ぶ。そして構造部分を数字で符号化し、テキスト部分はコンテナごとに L Z 7 7 などの方法で圧縮する。基本的にはパラメータ等の情報を必要とせず、アプリケーションのみで圧縮が可能である。必要であればパラメータ等を指定してコンテナごとの圧縮方法を指定し、圧縮効率を高めることが可能である。また、Cで実装されるため圧縮速度が速いという特徴をもつ。

## 【0007】

XML Z i p は、ルート要素からの深さを指定し、指定部分をドキュメント要素から分割し、残りをZ I Pで圧縮する。ルート要素部分は符号化されず直接操作することが可能になる。使用しない部分を圧縮することで文書へのアクセスを迅速に行える。ただし、圧縮効率はXM i l l に比較して低い。

## 【0008】

X C o m p は、XMLデータの構造部分のうち、D T D (Document Type Definition) から一意に決定される部分は符号化せず、一意に決定できない部分についての構造部分のみを圧縮する。テキスト部分はXM i l l と同じ方法で圧縮する。すなわち、以下の手順で圧縮を行う。（1）構造とコンテンツを分離する。（2）D T D からプッシュダウンオートマトン（P D A）を生成する。（3）生成されたP D Aを用いて構造部分を符号化する符号化トランスデューサを生成する。（4）符号化トランスデューサの各ノードに割り振られた数字をオートマトンを連鎖的に遷移することにより出力し、構造を符号化する。（5）得られた構造符号と要素ごとのコンテンツをL Z 7 7等で圧縮し、圧縮されたXML文書を出力する。

## 【0009】

上記XML文書に特化された圧縮手法のうち、X C o m p は構造部分の一部を符号化しないので、かなりよい圧縮効率を達成する。

## 【0010】

【発明が解決しようとする課題】

上記の通り、XCompはXML文書の圧縮手法として相対的に優れる。しかし、本発明者の検討によればXCompにおいても、XMLデータがある特定の構造をしている場合に圧縮効率が悪くなるという課題がある。すなわち、要素に「？」オペレータや「\*」オペレータ（「+」オペレータを含む）が適用された場合である。

## 【0011】

「？」オペレータは、ある要素の子要素が0回もしくは1回出現する場合に、要素の宣言文において子要素に付されるオペレータである。XCompにおいて「？」オペレータが出現した場合、「？」オペレータは、ある状態から別の状態に遷移する複数の選択肢で表現される。選択肢にはインデクスが付与される。XCompの実行時には、選択された選択肢に付与されたインデクスが出力される。選択肢の数は、「？」オペレータの後に「？」オペレータなどがいくつ連続するかによる。例えば、「？」オペレータが $n$ 個連続する場合に、最初の「？」オペレータに対する選択肢は $n+1$ 個となる。従って、インデクスは $n+1$ 種類必要であり、その1つを表現するには $O(\log n)$ ビット必要である。「？」オペレータが適用された要素がすべて存在すると、インデクスが複数羅列される。その1つを表現するのに $O(\log n)$ ビット必要であるから、それ全体を表現するには $O(n \log n)$ ビット必要である。

## 【0012】

「\*」オペレータは、ある要素の子要素が0回以上出現する場合に、要素の宣言文において子要素に付されるオペレータであり、「+」オペレータは、ある要素の子要素が1回以上出現する場合に、要素の宣言文において子要素に付されるオペレータである。XCompにおいて「\*」オペレータ（または「+」オペレータ）が出現した場合、「\*（または+）」オペレータは、同じ状態か別の状態に遷移する2つの選択肢で表現される。選択肢には、インデクスが付与される。XCompの実行時には、選択された選択肢に付与されたインデクスが出力される。「\*」オペレータが適用された要素が複数存在すると、同じインデクスが複数羅列される。インデクスの数は、存在する要素の数に比例する。従って、インデクス全体を表現するのに、要素の数を $n$ とすると、 $O(n)$ ビット必要である。

## 【 0 0 1 3 】

すなわち、X C o m p では D T D から一意的に決まらない構造のうち特定のものについて、それらを表現する符号のビット数が大きくなって、必ずしも十分な圧縮効率を達成できない問題がある。

## 【 0 0 1 4 】

本発明の目的は、より圧縮効率の高いXMLデータ（XML文書）の符号化方式を提案し、それを実現する方法およびシステムを提供することにある。

## 【 0 0 1 5 】

## 【課題を解決するための手段】

より高いXMLデータの圧縮効率を達成するために、本発明では、「ITU-T. X.680 - Abstract Syntax Notation One (ASN.1): Specification of basic notation, 1997. <http://www.itu.int/itudoc/itu-t/rec/x/x500up/x680.html>」に記載のASN.1を利用する。すなわち、符号化対象のXMLデータを構造とコンテンツ（テキスト）に分離する。一方、符号化対象のXMLデータの文法をASN.1抽象構文（型）に変換する。分離したコンテンツは要素ごとあるいはまとめてXM i l lと同様に圧縮する。また、分離した構造（要素）をASN.1抽象構文（型）に従うASN.1抽象構文（値）に変換する。次に、ASN.1抽象構文（値）をASN.1で規定されている符号化規則を用いてASN.1転送構文に変換する。ASN.1で規定される符号化規則には、BER、DER、PER等があるが、特に符号化効率の観点からPERを用いることが好ましい。そして符号化された要素（構文）であるASN.1転送構文と圧縮されたコンテンツ（テキスト）を併合し、符号化されたXMLデータを生成する。なお、PERについては、「ITU-T. X.691-ASN.1 encoding rules: Specification of Packed Encoding Rules (PER), 1997. <http://www.itu.int/itudoc/itu-t/rec/x/x500up/x691.html>」に詳しく記載されている。

## 【 0 0 1 6 】

このような符号化手法を用いることにより、X C o m p に比較して圧縮効率の高い符号化が行える。たとえば「？」オペレータがn個連続して出現した場合の



構造の符号化では、本発明を用いることにより $O(n)$ ビットでそれを表現できる。また、「\*」オペレータが出現した場合の構造の符号化では、本発明を用いることによりオーダとしてはXCompと同じであるが、実質的には少ないビット数でそれを表現できる。そして符号化（圧縮）されたXMLデータを通信あるいはストレージに用いることにより通信負荷を低減し、またストレージの容量を節約できる。

## 【0017】

なお、符号化されたXMLデータの復号化は、前記と逆の処理を行うことにより実現できる。また、本発明の符号化あるいは復号化の方法は、システムとして把握することも可能であり、さらに、前記方法によって実現される機能をコンピュータに実現させるためのプログラムとして把握することも可能である。

## 【0018】

## 【発明の実施の形態】

以下、本発明の実施の形態を図面を用いて詳細に説明する。ただし、本発明は多くの異なる態様で実施することが可能であり、本実施の形態の記載内容に限定して解釈すべきではない。なお、実施の形態の全体を通して同じ要素には同じ番号を付するものとする。

## 【0019】

以下の実施の形態では、主に方法またはシステムについて説明するが、当業者であれば明らかとなおり、本発明はコンピュータで使用可能なプログラムとしても実施できる。したがって、本発明は、ハードウェアとしての実施形態、ソフトウェアとしての実施形態またはソフトウェアとハードウェアとの組合せの実施形態をとることができる。プログラムは、ハードディスク、CD-ROM、光記憶装置または磁気記憶装置等の任意のコンピュータ可読媒体に記録できる。

## 【0020】

また以下の実施の形態では、一般的なコンピュータシステムを用いることができる。実施の形態で用いることができるコンピュータシステムは、中央演算処理装置（CPU）、主記憶装置（メインメモリ：RAM）、不揮発性記憶装置（ROM）、コプロセッサ、画像アクセラレータ、キャッシュメモリ、入出力制御装

置（I/O）等、一般的にコンピュータシステムに備えられるハードウェア資源を備える。また、ハードディスク装置等の外部記憶装置、インターネット等のネットワークに接続可能な通信手段を備えることができる。コンピュータシステムには、パーソナルコンピュータ、ワークステーション、メインフレームコンピュータ等各種のコンピュータが含まれる。

## 【0021】

### （実施の形態1）

#### 1. 前提条件

本実施の形態を説明するにあたり、本実施の形態における前提となる条件を述べる。

（1）XMLデータは、文法で構造が定義される。また、文法は、XMLデータを符号化する側と復号化する側で共有される。たとえば文法定義ファイルを外部データとして所定のIPアドレスに記録し、符号化あるいは復号化する際にIPアドレスを用いてこの外部ファイルを参照できる。なお、本実施の形態では文法定義としてDTDを用いるが、これに限られない。例えば、XML SchemaやRELAXなどを用いることができる。

（2）XMLデータは、要素とテキストだけから構成される。その他のもの（例えば、属性や処理命令）は、別の方法で管理される。例えば、属性や処理命令を特別な要素として表現し、XMLデータの中に埋め込むことができる。この際、文法も変更する必要がある。また、XMLデータから分離して、XPointerなどと共に別に格納することも考えられる。

（3）テキストは、XMILLなどの方法で圧縮される。

このような前提条件の下に本実施の形態のXMLデータの符号化および復号化が行われることを説明する。

## 【0022】

#### 2. 符号化手順

##### 2. 1 システム構成と手順の概要

図1は本実施の形態の符号化システムの一例をその機能について示したブロック図である。本実施の形態の符号化システムは、構文（型）生成装置2-1、分

離装置 2-2、構文（値）生成装置 2-3、転送構文生成装置 2-4、圧縮装置 2-5、併合装置 2-6 を有する。

#### 【 0 0 2 3 】

構文（型）生成装置 2-1 は、DTD から ASN. 1 抽象構文（型）を生成し、分離装置 2-2 は、XML データから要素の内容（テキスト）と構造（要素名および構造）を分離する。構文（値）生成装置 2-3 は、要素の構造から ASN. 1 抽象構文（値）を生成し、転送構文生成装置 2-4 は、ASN. 1 転送構文を生成する。圧縮装置 2-5 は分離したテキストを圧縮し、併合装置 2-6 は圧縮テキストと ASN. 1 転送構文を併合して符号化された XML データを生成する。

#### 【 0 0 2 4 】

このような符号化システムを用いた本実施の形態の符号化方法の手順の概要は以下のとおりである。

（ステップ 2-1） 構文（型）生成装置 2-1 を用いて DTD を ASN. 1 抽象構文（型）に変換する。

（ステップ 2-2） 分離装置 2-2 を用いてステップ 2-1 の DTD に従う XML データからテキストを分離する。なお、ステップ 2-2 は、ステップ 2-1 と並行に行われても良く、ステップ 2-1 よりも先に行われても良い。

（ステップ 2-3） 構文（値）生成装置 2-3 を用いてステップ 2-2 でテキストが分離された後の XML データ（要素の構文）を、ステップ 2-1 の ASN. 1 抽象構文（型）に従う ASN. 1 抽象構文（値）に変換する。

（ステップ 2-4） 転送構文生成装置 2-4 を用いてステップ 2-3 の ASN. 1 抽象構文（値）を ASN. 1 転送構文に変換する。

（ステップ 2-5） 圧縮装置 2-5 を用いてステップ 2-2 で分離されたテキストを圧縮する。なお、ステップ 2-5 はステップ 2-3 ～ステップ 2-4 と並行に行われる。

（ステップ 2-6） 併合装置 2-6 を用いてステップ 2-4 で生成された ASN. 1 転送構文とステップ 2-5 で生成された圧縮テキストを併合し、符号化 XML データを生成する。

## 【0025】

なお、ASN. 1 抽象構文（値）をASN. 1 転送構文に変換するには、ASN. 1 で規定されている符号化規則に従う。そのような規則には、BERやDER、PERなどがある。特に、PERは、ASN. 1 抽象構文（型）から一意に決定される型やその値などを符号化しないため、符号化効率がよい。

## 【0026】

以下、前記各ステップを詳細に説明する。

## 2. 2 ステップ2-1

ステップ2-1では、DTDをASN. 1 抽象構文（型）に変換する。以下では、内容モデルのパターンごとに、その方法を説明する。

## 【0027】

## 2. 2. 1 要素内容(element content)

XMLの要素内容は、要素名とオペレータの組み合わせで構成される。通常要素名は文字列で表される。オペレータは要素内容における子要素の出現順と出現回数を指定するための演算子である。周知のとおりXMLでは、「,」、「|」、「?」、「\*」、「+」の各オペレータが許可される。

原則として、要素名はASN. 1 抽象構文では識別子として、オペレータは同じくASN. 1 抽象構文では型として表現する。ただし、識別子はBERなどでは符号化されないため、要素名を識別子とする必然性はない。以下では、オペレータごとに、それをどのような型で表現するかを説明する。

## 【0028】

## 2. 2. 1. 1 「,」オペレータ

「,」オペレータは、ASN. 1 抽象構文ではsequence型で表現する。例えば、DTDとして、

```
<!ELEMENT a (b,c)>
```

```
<!ELEMENT b (#PCDATA)>
```

```
<!ELEMENT c (#PCDATA)>
```

が与えられるとき、ASN. 1 抽象構文（型）は、

```
A ::= SEQUENCE {
```

```

    b   B、
    c   C }
B ::= NULL
C ::= NULL

```

となる。ここで、A、B、Cは、便宜上導入した型参照である。型参照は、衝突さえしなければどのようなものでもよい。また、テキストはステップ2-2で分離されるため、BやCはnull型とする。

【0029】

## 2. 2. 1. 2 「|」オペレータ

「|」オペレータは、ASN. 1抽象構文ではchoice型で表現する。例えば、DTDとして、

```

<!ELEMENT a (b|c)>
<!ELEMENT b (#PCDATA)>
<!ELEMENT c (#PCDATA)>

```

が与えられているとき、ASN. 1抽象構文(型)は、

```

A ::= CHOICE {
    b   B、
    c   C }
B ::= NULL
C ::= NULL

```

となる。

【0030】

## 2. 2. 1. 3 「?」オペレータ

「?」オペレータは、ASN. 1抽象構文ではsequence型とキーワード「OPTIONAL」の組み合わせで表現する。例えば、DTDとして、

```

<!ELEMENT a (b?)>
<!ELEMENT b (#PCDATA)>

```

が与えられるとき、ASN. 1抽象構文(型)は、

```

A ::= SEQUENCE {

```

b B OPTIONAL }

B ::= NULL

となる。

【0031】

2. 2. 1. 4 「\*」オペレータ

「\*」オペレータは、ASN. 1 抽象構文では `sequence-of` 型で表現する。例えば、DTD として、

<!ELEMENT a (b\*)>

<!ELEMENT b (#PCDATA)>

が与えられるとき、ASN. 1 抽象構文 (型) は、

A ::= SEQUENCE OF B

B ::= NULL

となる。

【0032】

2. 2. 1. 5 「+」オペレータ

「+」オペレータは、ASN. 1 抽象構文ではサイズが制限された `sequence-of` 型で表現する。ただし、サイズ制限は符号化には影響しないため、「+」オペレータに対する符号は「\*」オペレータに対するそれと同じものになる。例えば、DTD として、

<!ELEMENT a (b+)>

<!ELEMENT b (#PCDATA)>

が与えられるとき、ASN. 1 抽象構文 (型) は、

A ::= SEQUENCE SIZE (1 .. MAX) OF B

B ::= NULL

となる。

【0033】

2. 2. 1. 6 オペレータなし

要素内容には、オペレータの適用されていない要素名が1つだけ指定されることもある。そのような要素名は、ASN. 1 抽象構文では `defined` 型で表

現する。例えば、DTDとして、

```
<!ELEMENT a (b)>
```

```
<!ELEMENT b (#PCDATA)>
```

が与えられるとき、ASN. 1 抽象構文 (型) は、

```
A ::= B
```

```
B ::= NULL
```

となる。

【0034】

## 2. 2. 2 混在内容(mixed content)

混在内容は、キーワード「#PCDATA」と1つ以上の要素名を「|」オペレータで結合した後、「\*」オペレータを適用したものである。そこで、ASN. 1 抽象構文ではchoice型とsequence-of型の組み合わせで表現する。例えば、DTDとして、

```
<!ELEMENT a (#PCDATA|b)*>
```

```
<!ELEMENT b (#PCDATA)>
```

が与えられるとき、ASN. 1 抽象構文 (型) は、

```
A ::= SEQUENCE OF CHOICE {
```

```
    txt    NULL,
```

```
    b      B }
```

```
B ::= NULL
```

となる。ここで、txtは便宜上導入した識別子であり、混在内容に含まれるテキストに対応する。

【0035】

## 2. 2. 3 空要素(EMPTY)

空要素は、ASN. 1 抽象構文ではnull型で表現する。例えば、DTDが

```
<!ELEMENT a EMPTY>
```

の場合はASN. 1 抽象構文 (型) は、

```
A ::= NULL
```

となる。

#### 【0036】

#### 2. 2. 4 任意要素 (ANY)

任意要素は、キーワード「#PCDATA」とDTDで宣言されたすべての要素名から構成される混在内容と等価である。従って、任意要素のASN. 1抽象構文における表現は、混在内容のそれに帰着される。

#### 【0037】

#### 2. 3 ステップ2-2

ステップ2-2では、ステップ2-1のDTDに従うXMLデータからテキストを分離する。例えば、XMLデータが、

```
<a>
  <b>10</b>
  <c>20</c>
</a>
```

の場合、要素bから「10」が、要素cから「20」が分離される。その結果、XMLデータ（要素名および構造）は次のようになる。

```
<a>
  <b />
  <c />
</a>
```

#### 【0038】

分離されたテキストを圧縮する方法は、例えば、XM111などのように、要素ごとにまとめた後、圧縮することができる。なお、その他の圧縮方法を適用してももちろん良い。

#### 【0039】

#### 2. 4 ステップ2-3

ステップ2-3では、ステップ2-2のXMLデータを、ステップ2-1のASN. 1抽象構文（型）に従うASN. 1抽象構文（値）に変換する。例えば、ASN. 1抽象構文（型）が、



```
A ::= SEQUENCE {
```

```
  b  B,
```

```
  c  C }
```

```
B ::= NULL
```

```
C ::= NULL
```

の場合、以下のXMLデータ（要素名および構造）、

```
<a>
```

```
  <b />
```

```
  <c />
```

```
</a>
```

をASN. 1 抽象構文（値）に変換すると、

```
a A ::= {
```

```
  b  NULL,
```

```
  c  NULL }
```

となる。

【0040】

## 2. 5 ステップ2-4

ステップ2-4では、ASN. 1で規定されている符号化規則に従って、ステップ2-3のASN. 1 抽象構文（値）をASN. 1 転送構文に変換する。そのような規則にはBERやDER、PER（ALIGNED/UNALIGNED）などがある。しかしながら、符号化効率を向上する観点からPER（UNALIGNED）を利用することが好ましい。尤も、BER、DER、PER（ALIGNED）を利用してもよいことは勿論である。

【0041】

## 2. 6 ステップ2-5

ステップ2-5では、テキストを圧縮する。圧縮の具体的方法については、周知のLZ77等を用いる。その他、従来技術の項で説明した公知技術を用いることができる。圧縮は、要素ごとに行われても良く、また、各要素をまとめて圧縮しても良い。

## 【 0 0 4 2 】

## 2. 7 ステップ 2 - 6

ステップ 2 - 6 では、ASN. 1 転送構文と圧縮テキストを併合する。両データを単に結合することも可能であるが、復号時の分離を考慮したセパレータをデータ間に挿入し、あるいはデータビット数情報を持つヘッダ等を付加しても良い。

## 【 0 0 4 3 】

## 3. 復号化手順

## 3. 1 システム構成と手順の概要

図 2 は本実施の形態の復号化システムの一例をその機能について示したブロック図である。本実施の形態の復号化システムは、構文（型）生成装置 3 - 1、転送構文復号装置 3 - 2、抽象構文復号装置 3 - 3、併合装置 3 - 4、分離装置 3 - 5、解凍装置 3 - 6 を有する。

## 【 0 0 4 4 】

構文（型）生成装置 3 - 1 は、前記した構文（型）生成装置 2 - 1 と同様に、DTD を ASN. 1 抽象構文（型）に変換する。分離装置 3 - 5 は、符号化された XML データから ASN. 1 転送構文と圧縮テキストを分離し、転送構文復号装置 3 - 2 は、ASN. 1 転送構文を ASN. 1 抽象構文（型）に従う ASN. 1 抽象構文（値）に変換する。抽象構文復号装置 3 - 3 は、ASN. 1 抽象構文（値）を DTD に従う XML データ（要素名と構造）に変換する。併合装置 3 - 4 は復号化されたテキスト（要素の内容）と XML データ（要素名と構造）を併合し、XML データを生成する。解凍装置 3 - 6 は、圧縮テキストを解凍する。

## 【 0 0 4 5 】

このような復号化システムを用いた本実施の形態の復号化方法の手順は、前記符号化の手順をほぼ逆に行う。その概要は以下のとおりである。

（ステップ 3 - 1） DTD を ASN. 1 抽象構文（型）に変換する。

（ステップ 3 - 2） 符号化 XML データを圧縮テキストと ASN. 1 転送構文に分離する。なお、ステップ 3 - 2 は、ステップ 3 - 1 と並行に行われても良く、ステップ 3 - 1 よりも先に行われても良い。

(ステップ3-3) ASN. 1 転送構文を、ステップ3-1のASN. 1 抽象構文(型)に従うASN. 1 抽象構文(値)に変換する。

(ステップ3-4) ステップ3-3のASN. 1 抽象構文(値)を、ステップ3-1のDTDに従うXMLデータ(要素名と構造)に変換する。

(ステップ3-5) ステップ3-2で分離した圧縮テキストを解凍する。

(ステップ3-6) ステップ3-4のXMLデータ(要素名と構造)にステップ3-5で解凍したテキストを結合する。

#### 【0046】

なお、前記ステップ3-1～ステップ3-6における復号化の各処理は、それに相当する符号化の場合の逆であって自明である。よって、その詳細な説明を省略する。

以下、DTDの変換およびそのDTDに従うXMLデータのPERによる符号化の具体例を示す。

#### 【0047】

#### 4. 「,」オペレータを含む場合

ここでは、「,」オペレータを含むDTDと、それに従うXMLデータを例にとる。以下のDTD、

```
<!ELEMENT a (b,c)>
```

```
<!ELEMENT b (#PCDATA)>
```

```
<!ELEMENT c (#PCDATA)>
```

が与えられている場合、前記ステップ2-1で生成されるASN. 1 抽象構文(型)は、

```
A ::= SEQUENCE {
```

```
  b  B,
```

```
  c  C }
```

```
B ::= NULL
```

```
C ::= NULL
```

となる。このようなDTDに従う符号化対象のXMLデータとして、

```
<a>
```

<b>10</b>

<c>20</c>

</a>

を例示すれば、前記ステップ 2 - 2 で分離される要素（XML データの要素名と構造）は、

<a>

<b />

<c />

</a>

となる。この要素から前記ステップ 2 - 3 で生成される ASN. 1 抽象構文（値）は、

```
a A ::= {
    b  NULL,
    c  NULL }
```

となる。

【 0 0 4 8 】

PER では、sequence 型の値は、原則としてその構成要素の値がその順番で符号化される。しかし、ここでは b や c の値は null であり、null は空ビット列に符号化されるため、a の値は空ビット列となる。その場合に、符号は例外的に、

00000000<sub>(2)</sub>

のようになる。このようにして、ASN. 1 転送構文が生成される。なお、このような ASN. 1 転送構文の生成は前記ステップ 2 - 4 で行われる。また、下付き文字の “ ( 2 ) ” は 2 進数であることを示す。

【 0 0 4 9 】

その後、前記ステップ 2 - 5 で圧縮処理された圧縮テキストと ASN. 1 転送構文（ここでは「00000000<sub>(2)</sub>」）が前記ステップ 2 - 6 で併合され、符号化 XML データが生成される。

【 0 0 5 0 】

## 5. 「|」オペレータを含む場合

ここでは、「|」オペレータを含むDTDと、それに従うXMLデータを例にとる。以下のDTD、

```
<!ELEMENT a (b|c)>
<!ELEMENT b (#PCDATA)>
<!ELEMENT c(#PCDATA)>
```

が与えられている場合、前記ステップ2-1で生成されるASN. 1抽象構文(型)は、

```
A ::= CHOICE {
    b   B,
    c   C }
B ::= NULL
C ::= NULL
```

となる。このようなDTDに従う符号化対象のXMLデータとして、

```
<a>
  <b>10</b>
</a>
```

を例示すれば、前記ステップ2-2で分離される要素(XMLデータの要素名と構造)は、

```
<a>
  <b />
</a>
```

となる。この要素から前記ステップ2-3で生成されるASN. 1抽象構文(値)は、

```
a A ::= b:NULL
```

となる。

```
【0051】
```

PERでは、choice型の値は、まず選択された構成要素のインデクス(0ベース)が符号化され、次にその構成要素の値が符号化される。ここでは b

が選択されているため、インデクスは0である。また、bの値はnull である。従って、aの値は、

0XXXXXXX<sub>(2)</sub>

のようになる。ここで、Xは、8ビットの倍数にするために付加されたパディング・ビットを表わす。このようにして、ASN. 1 転送構文が生成される。なお、圧縮テキストの生成および圧縮テキストとASN. 1 転送構文の併合は前記“4. 「,」オペレータを含む場合”と同様である。

【0052】

#### 6. 「？」オペレータを含む場合

ここでは、「？」オペレータを含むDTDと、それに従うXMLデータを例にとる。以下のDTD、

```
<!ELEMENT a (b?,c)>
```

```
<!ELEMENT b (#PCDATA)>
```

```
<!ELEMENT c (#PCDATA)>
```

が与えられている場合、前記ステップ2-1で生成されるASN. 1 抽象構文（型）は、

```
A ::= SEQUENCE {
    id0  SEQUENCE {
        b  B OPTIONAL },
    c    C }
```

```
B ::= NULL
```

```
C ::= NULL
```

となる。このようなDTDに従う符号化対象のXMLデータとして、

```
<a>
  <b>10</b>
  <c>20</c>
</a>
```

を例示すれば、前記ステップ2-2で分離される要素（XMLデータの要素名と構造）は、

```

<a>
  <b />
  <c />
</a>

```

となる。この要素から前記ステップ2-3で生成されるASN. 1抽象構文（値）は、

```

a A ::= {
  id0 {
    b NULL }
    c NULL }

```

となる。

【0053】

PERでは、sequence型の構成要素が少なくとも1つ「OPTIONAL」と指定されている場合に、各構成要素の値が符号化される前に、どの構成要素が存在しているかを表わすビット列が付加される。ビットは、構成要素が存在する場合に1、存在しない場合に0となる。ここではbだけが「OPTIONAL」と指定されており、それが存在しているため、まずビット列1が付加される。次にbとcの値が符号化されるが、それらは共にnullであるため、aの値は、

1XXXXXX<sub>(2)</sub>

のようになる。このようにして、ASN. 1転送構文が生成される。なお、圧縮テキストの生成および圧縮テキストとASN. 1転送構文の併合は前記“4. 「,」オペレータを含む場」と同様である。

【0054】

#### 7. 「\*」オペレータを含む場合

ここでは、「\*」オペレータを含むDTDと、それに従うXMLデータを例にとる。以下のDTD、

```

<!ELEMENT a (b*)>
<!ELEMENT b (#PCDATA)>

```

が与えられている場合、前記ステップ2-1で生成されるASN. 1抽象構文（型）は、

```
A ::= SEQUENCE OF B
```

```
B ::= NULL
```

となる。このようなDTDに従う符号化対象のXMLデータとして、

```
<a>
  <b>10</b>
  <b>20</b>
</a>
```

を例示すれば、前記ステップ2-2で分離される要素（XMLデータの要素名と構造）は、

```
<a>
  <b />
  <b />
</a>
```

となる。この要素から前記ステップ2-3で生成されるASN. 1抽象構文（値）は、

```
a A ::= {
  NULL,
  NULL }
```

となる。

【0055】

PERでは、sequence-of型の値は、まずその構成要素の数が符号化され、次に各構成要素の値がその順番で符号化される。ここでは構成要素の数は2であり、まずその値が符号化される。次に各構成要素の値が符号化されるが、それらはすべてnullであるため、aの値は、

00000010<sub>(2)</sub>

のようになる。このようにして、ASN. 1転送構文が生成される。なお、圧縮テキストの生成および圧縮テキストとASN. 1転送構文の併合は前記“4. 「



、」オペレータを含む場合”と同様である。

【0056】

#### 8. 混在内容を含む場合

ここでは、混在内容を含むDTDと、それに従うXMLデータを例にとる。以下のDTD、

```
<!ELEMENT a (#PCDATA|b)*>
```

```
<!ELEMENT b (#PCDATA)>
```

が与えられている場合、前記ステップ2-1で生成されるASN. 1抽象構文(型)は、

```
A ::= SEQUENCE OF CHOICE {
```

```
    txt    NULL,
```

```
    b      NULL }
```

となる。このようなDTDに従う符号化対象のXMLデータとして、

```
<a>
```

```
    xxx
```

```
    <b>10</b>
```

```
    <b>20</b>
```

```
</a>
```

を例示すれば、前記ステップ2-2で分離される要素(XMLデータの要素名と構造)は、

```
<a>
```

```
    <txt />
```

```
    <b />
```

```
    <b />
```

```
</a>
```

となる。

【0057】

なお、上記において「xxx」はテキストの内容を示し、「txt」は要素がテキストであることを示す。この要素から前記ステップ2-3で生成されるASN

N. 1 抽象構文 (値) は、

```
a A ::= {
    txt:NULL,
    b:NULL,
    b:NULL }
```

となる。

【0058】

sequence-of型の値とchoice型の値をPERで符号化する方法は、前述の通りである。ここでは、構成要素の数は3である。また、構成要素としてtxt、b、bがその順番で選択されているため、インデクスは0、1、1となる。それらの値はすべてnullであるため、aの値は、

00000011<sub>(2)</sub>

011XXXXX<sub>(2)</sub>

のようになる。このようにして、ASN. 1 転送構文が生成される。なお、圧縮テキストの生成および圧縮テキストとASN. 1 転送構文の併合は前記“4. 「,」オペレータを含む場合”と同様である。

【0059】

(実施の形態2)

前記実施の形態1では、XMLデータは要素とテキストだけから構成され、その他のもの(例えば、属性や処理命令)は別の方法で管理されると仮定して説明を行った。本実施の形態では、属性や処理命令等が含まれる場合の処理の一例を説明する。すなわち、それら属性等を特別な要素で表現し、XMLデータの中に埋め込む方法を例示する。この場合、文法も変更する必要がある。なお、本実施の形態の例に関わらず、属性等をXMLデータから分離して、XPointerなどと共に別に格納する方法を採用することも可能である。このような属性等を特別な要素としてXMLデータに埋め込む方策は、前記実施の形態1の処理の前処理および後処理として把握することが可能である。これにより広い範囲のXMLデータを扱えるようになる。以下では、そのような前処理の例として、DTDで定義される項目をXMLデータの中に埋め込む方法を説明する。

## 【 0 0 6 0 】

## 9. 1 前処理システムと前処理の概要

図 3 は、本実施の形態の前処理システムの一例をその機能について示したブロック図である。本実施の形態の前処理システムは、DTD 変換装置 9-1、XML データ変換装置 9-2、符号化装置 9-3 を有する。

## 【 0 0 6 1 】

DTD 変換装置 9-1 は、一般の DTD を要素以外の項目についても考慮した DTD' に変換する。XML データ変換装置 9-2 は、一般の DTD に従う XML データを DTD' に従う XML データ' に変換する。符号化装置 9-3 は、実施の形態 1 の符号化システムである。

## 【 0 0 6 2 】

本実施の形態の前処理方法は、装置 9-1 による DTD の変換および装置 9-2 による XML データの変換の各ステップを有する。各々変換後、実施の形態 1 の符号化処理を行うことにより符号化 XML データが生成される。

## 【 0 0 6 3 】

## 9. 2 DTD の変換処理

DTD には、要素型宣言以外に属性リスト宣言、エンティティ宣言、記法宣言の 3 つの宣言が含まれうる。

## 【 0 0 6 4 】

エンティティ宣言で定義されるエンティティには、パース対象エンティティ、パース対象外エンティティ、パラメータ・エンティティがある。パース対象エンティティは、テキストや属性値など、あらゆる場所で参照されうる。このエンティティは、単純に展開する。また、パース対象外エンティティは、属性値でしか参照されえない。従って、属性を処理できれば十分である。一方、パラメータ・エンティティは、DTD の中でしか参照されえない。従って、パラメータ・エンティティは本実施の形態では考慮しない。

## 【 0 0 6 5 】

記法宣言で定義される記法は、属性値でしか参照されえない。従って、属性を処理できれば十分である。

## 【0066】

以上の議論から、DTDに対する前処理は、要素型宣言と属性リスト宣言に対するそれに帰着される。

## 【0067】

属性リスト宣言では、ある要素に付与される属性と、その属性の取りうる値が定義される。また、必要であれば、デフォルト値も定義される。属性は、「REQUIRED」であるか、「IMPLIED」であるか、デフォルト値が定義されているか、それは「FIXED」であるかで、要素に対する付与のされ方が変わる。従って、属性は原則として要素で表現することにするが、その定義に従って表現を多少変えることが好ましい。

## 【0068】

属性リスト宣言に従って要素型宣言を変更する規則は、以下のとおりである。

(1) 属性は、要素（以下、属性要素）で表現する。要素名は、属性名から一意に決定できるようにする。例えば、“親の要素名” + “\_” + “属性名”のようにする。ただし、要素名がすでに使用されていないことに注意する。以下では、この命名規則に従うことにする。

(2) 属性値はすべて「CDATA」として扱い、属性要素に含める。

(3) 属性要素は、親要素の子要素の先頭に挿入する。

(4) 「REQUIRED」属性は、要素で表現する。

(5) 「IMPLIED」属性は、「？」オペレータが適用された要素で表現する。

(6) デフォルト値が定義されている属性は、「？」オペレータが適用された要素で表現する。

(7) 「FIXED」デフォルト値が定義されている属性は、無視する。

## 【0069】

以下具体例をあげる。例えば、DTDとして、

```
<!ELEMENT a (b,c)>
```

```
<!ATTLIST a w CDATA #REQUIRED
```

```
      x ID #IMPLIED
```

y (0|1|2) "0"

z CDATA #FIXED "abc" >

が与えられたとき、DTD' は、

<!ELEMENT a (a\_w,a\_x?,a\_y?,b,c)>

<!ELEMENT a\_w (#PCDATA)>

<!ELEMENT a\_x (#PCDATA)>

<!ELEMENT a\_y (#PCDATA)>

となる。

【0070】

### 9. 3 XMLデータの変換処理

ここでは、上記のDTDに従うXMLデータを上記DTD' に従うXMLデータ' に変換する。基本的には、要素に付与されている属性を属性要素に変換し、その属性要素をその要素の子要素の先頭に挿入すればよい。ただし、属性にデフォルト値が定義されており、属性値がそれと一致するのであれば、属性要素には変換しない。

【0071】

例えば、9. 2におけるDTDに従うXMLデータとして、

<a w="xyz" y="0" z="abc" >

<b>...</b>

<c>...</c>

</a>

を例示すれば、XMLデータ' は、

<a>

<a\_w>xyz</a\_w>

<b>...</b>

<c>...</c>

</a>

に変換される。ここで、a要素にはx属性が付与されていないため、a\_\_x要素は現れていない。また、y属性の値はデフォルト値と一致するため、a\_\_y要素

も現れていない。さらに、z属性は「FIXED」デフォルト値が定義されているため、削除されている。

【0072】

#### 9. 4 後処理システムと後処理の概要

図4は、本実施の形態の後処理システムの一例をその機能について示したブロック図である。本実施の形態の後処理システムは、DTD変換装置9-4、復号化装置9-5、XMLデータ変換装置9-6を有する。

【0073】

DTD変換装置9-4はDTD変換装置9-1と同様である。復号化装置9-5は、実施の形態1の復号化システムである。XMLデータ変換装置9-6は、DTD'に従うXMLデータ'からDTDに従うXMLデータに変換する。

【0074】

本実施の形態の後処理方法は、装置9-4によるDTDの変換および装置9-6によるXMLデータの変換の各ステップを有する。実施の形態1の復号化処理の後、これらステップを実行する。装置9-4によるDTDの変換は9.2と同様であり、装置9-6によるXMLデータの変換は9.3の変換を逆に行うことにより実行できる。よって、詳細な説明は省略する。なお、図中破線矢印は、必要があればDTDを参照することを意味する。属性要素を識別でき、かつその要素名から属性名を一意に決定できるのであれば、DTDを参照する必要はない。また、後処理は前処理がいかに行われたかに依存するので、前記前処理が異なる場合に本実施の形態の後処理もそれに併せて変更されることは勿論である。

【0075】

#### 10. 実施の形態の効果

上記した実施の形態1、2の符号化方法を用いれば、XMLデータを効率良く圧縮することが可能である。以下、XCompとの比較において、本実施の形態の効果を説明する。

【0076】

XMLデータを符号化するとき、DTDから一意に決定される情報は符号化しないという点で、本手法とXCompは同じである。しかし、XCompは、

XMLデータがある特定の構造をしている場合に、圧縮効率が悪くなる。具体的には、要素に「？」オペレータや「\*」オペレータ（「+」オペレータを含む）が適用された場合である。

【0077】

#### 10.1 「？」オペレータ

本実施の形態の手法は要素が存在するかどうかをビット列で表現するのに対して、XCompは存在する要素（選択肢）に付与されたインデックスを羅列する。この差は、「？」オペレータが適用された要素がいくつか連続し、それらがすべて存在する場合に顕著に現れる。

【0078】

例えば、DTD：

<!ELEMENT a (b?,c?)>

<!ELEMENT b (#PCDATA)>

<!ELEMENT c (#PCDATA)>

に従うXMLデータ：

<a>

<b>xxx</b>

<c>yyy</c>

</a>

を考える。このXMLデータには、要素bと要素cが共に存在している。このXMLデータは、本手法では11(2)に符号化される。ただし、簡単のため、パディング・ビットは付加していない。

【0079】

一方、XCompでは、オートマトン：

	S1	S2	S3	S4
S1	-	b/1	c/2	$\epsilon$ /3
S2	-	-	c/1	$\epsilon$ /2
S3	-	-	-	$\epsilon$ / $\epsilon$
S4	-	-	-	-

が作成されたとすると、インデクス列 1 1 が得られる。このインデクス列を 0 ベースに変換し、必要最小限のビット数で符号化すると、符号 0 0 0 (2) が得られる。本手法による符号と X C o m p によるそれを比較すると、本手法のほうが 1 ビットだけ短く符号化できることが分かる。

## 【 0 0 8 0 】

表 1 に、「？」オペレータが適用された要素がいくつか連続し、それらがすべて存在する場合に、それらを符号化するのに必要なビット数を、本手法と X C o m p で比較したものを示す。表 1 から、本手法は X C o m p と同じか、それよりも効率よく符号化できることが分かる。一般には、「？」オペレータが適用された要素が  $n$  個連続する場合に、X C o m p は  $O(n \log n)$  ビットを必要とするのに対して、本手法は  $O(n)$  ビットで十分である。

## 【 0 0 8 1 】

【表 1】

" ? " の数	XComp(従来技術)			本発明
	選択肢数	ビット数	トータルビット数	ビット数
1	2	1	1	1
2	3、2	2、1	3	2
3	4、3、2	2、2、1	5	3
4	5、4、3、2	3、2、2、1	8	4
5	6、5、4、3、2	3、3、2、2、1	11	5

## 【 0 0 8 2 】

## 1 0 . 2 「\*」オペレータ

本実施の形態の手法は存在する要素の数を符号化するのに対して、X C o m p は存在する要素（選択肢）に付与されたインデクスを羅列する。この差は、「\*」オペレータが適用された要素がいくつも存在する場合に顕著に現れる。

## 【 0 0 8 3 】

例えば、DTD :

<!ELEMENT a (b\*)>

<!ELEMENT b (#PCDATA)>

に従う XML データ :

<a>



<b>xxx</b>

<b>yyy</b>

...

<b>zzz</b>

</a>

を考える。ここで、要素 b は 8 回出現しているとする。この XML データは、本手法では 0 0 0 0 1 0 0 0 (2) に符号化される。

【0084】

一方、X C o m p では、オートマトン：

	S0	S1
S0	b/1	$\varepsilon / 2$
S1	-	-

が作成されたとすると、インデクス列 1 1 1 1 1 1 1 2 が得られる。このインデクス列を 0 ベースに変換し、必要最小限のビット数で符号化すると、符号 0 0 0 0 0 0 0 1 (2) が得られる。本手法による符号と X C o m p によるそれとを比較すると、本手法のほうが 1 ビットだけ短く符号化できることが分かる。

【0085】

図 5 に、「\*」オペレータが適用された要素がいくつも存在する場合に、それらを符号化するのに必要なビット数を、本手法と X C o m p で比較したものを示す。同図から、要素が 8 つ以上存在する場合に、本手法は X C o m p よりも効率よく符号化できることが分かる。本手法は、一般には X C o m p と同じオーダのビット数を必要とするが、実質的にはそれよりも少ないビット数で十分である。

【0086】

以上、本発明者によってなされた発明を発明の実施の形態に基づき具体的に説明したが、本発明は前記実施の形態に限定されるものではなく、その要旨を逸脱しない範囲で種々変更可能である。

【0087】

たとえば、本発明の手法は、複数のオペレータを組み合わせ適用しても良い。内容モデルで使用されうるオペレータには、「,」、「|」、「?」、「\*」

(「+」を含む)の4つがある。従って、適用順序を考慮に入れると、それらの組み合わせは16通りある。各組み合わせに対して、前記実施の形態と同様にXMLデータをASN.1抽象構文に変換することができる。よって、本発明を利用してあらゆるオペレータの組合せを含むXMLデータを符号化できる。

## 【0088】

また、前記実施の形態で説明した手法のポイントの1つは、文法やそれに従うXMLデータのASN.1抽象構文における表現を定義したことである。しかし、それとは別の表現も考えられる。

## 【0089】

たとえば、「?」オペレータは、前記実施の形態ではsequence型とキーワード「OPTIONAL」の組み合わせで表現した。具体的には、DTD:

```
<!ELEMENT a (b?,c)>
```

は、ASN.1抽象構文:

```
A ::= SEQUENCE {
    id0 SEQUENCE {
        b B OPTIONAL },
    c C }
```

に変換される。

## 【0090】

しかし、先のDTDはASN.1抽象構文:

```
A ::= SEQUENCE {
    b B OPTIONAL,
    c C }
```

に変換されてもよい。このように変換されるほうが、BERやDERで符号化する場合に符号が短くなるため都合がよい。

## 【0091】

ただし、この方法では、以下のDTD:

```
<!ELEMENT a (b?)>
<!ELEMENT a (b?|c)>
```

のような場合にうまく機能しないので、実施の形態のように表現するほうが妥当である。

## 【 0 0 9 2 】

また、前記実施の形態では、オペレータは何からの型で表現される。例えば、「,」オペレータは `sequence` 型で表現され、「|」オペレータは `choice` 型で表現される。一方、オペレータの適用をプロダクションと見なし、すべて `sequence` 型で表現するという方法も採れる。その場合に、`sequence` 型で表現されていない「|」オペレータ、「\*」オペレータ（「+」オペレータを含む）およびオペレータなしは、次のように表現できる。

## 【 0 0 9 3 】

すなわち、`<!ELEMENT a(b|c)>`は、

```
A ::= SEQUENCE {
    id0 CHOICE {
        b B,
        c C } }
```

と表現できる。また、`<!ELEMENT a (b*)>`は、

```
A ::= SEQUENCE {
    b SEQUENCE OF B }
```

と表現できる。また、`<!ELEMENT a (b)>`は、

```
A ::= SEQUENCE {
    b B }
```

と表現できる。

## 【 0 0 9 4 】

この表現法では、`sequence` 型で統一的に表現することで、見た目には分かりやすくなる。また、オペレータを `sequence` 型でラップすることに相当するので、複数のオペレータを組み合わせてもうまく機能することが分かる。ただし、BERやDERでは常に型が符号化されるため、型を挿入すればするほど符号は長くなる。従って、必要以上の型を挿入するべきではなく、その理由から前記実施の形態の手法で表現するほうが妥当である。

【 0 0 9 5 】

【発明の効果】

本願で開示される発明のうち、代表的なものによって得られる効果は、以下の通りである。すなわち、XMLデータの符号化（圧縮）効率を高めることが可能になり、また、属性等要素以外の記述を含めたXMLデータの符号化（圧縮）が可能になる。これにより、XMLデータの転送による通信負荷を軽減し、また、XMLデータのストレージ容量を少なくできる。

【図面の簡単な説明】

【図 1】

本発明の一実施の形態である符号化システムの一例をその機能について示したブロック図である。

【図 2】

本発明の一実施の形態である復号化システムの一例をその機能について示したブロック図である。

【図 3】

本発明の一実施の形態である前処理システムの一例をその機能について示したブロック図である。

【図 4】

本発明の一実施の形態である後処理システムの一例をその機能について示したブロック図である。

【図 5】

本実施の形態の符号化手法の符号化率をXC o m p との比較において示した図である。

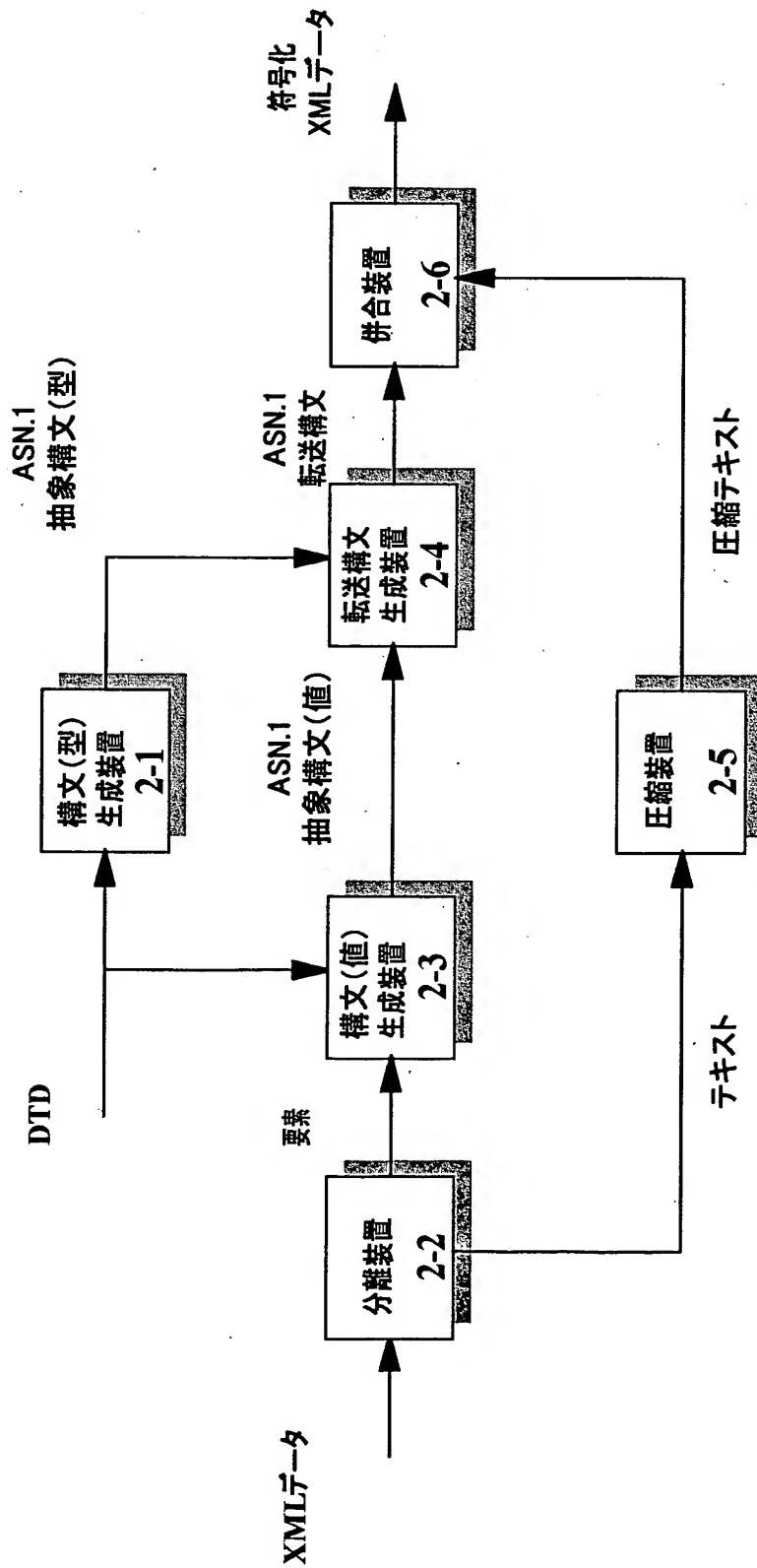
【符号の説明】

2 - 1 … 構文（型）生成装置、2 - 2 … 分離装置、2 - 3 … 構文（値）生成装置、2 - 4 … 転送構文生成装置、2 - 5 … 圧縮装置、2 - 6 … 併合装置、3 - 1 … 構文（型）生成装置、3 - 2 … 転送構文復号装置、3 - 3 … 抽象構文復号装置、3 - 4 … 併合装置、3 - 5 … 分離装置、3 - 6 … 解凍装置、9 - 1 … DTD変換装置、9 - 2 … XMLデータ変換装置、9 - 3 … 符号化装置、9 - 4 … DTD

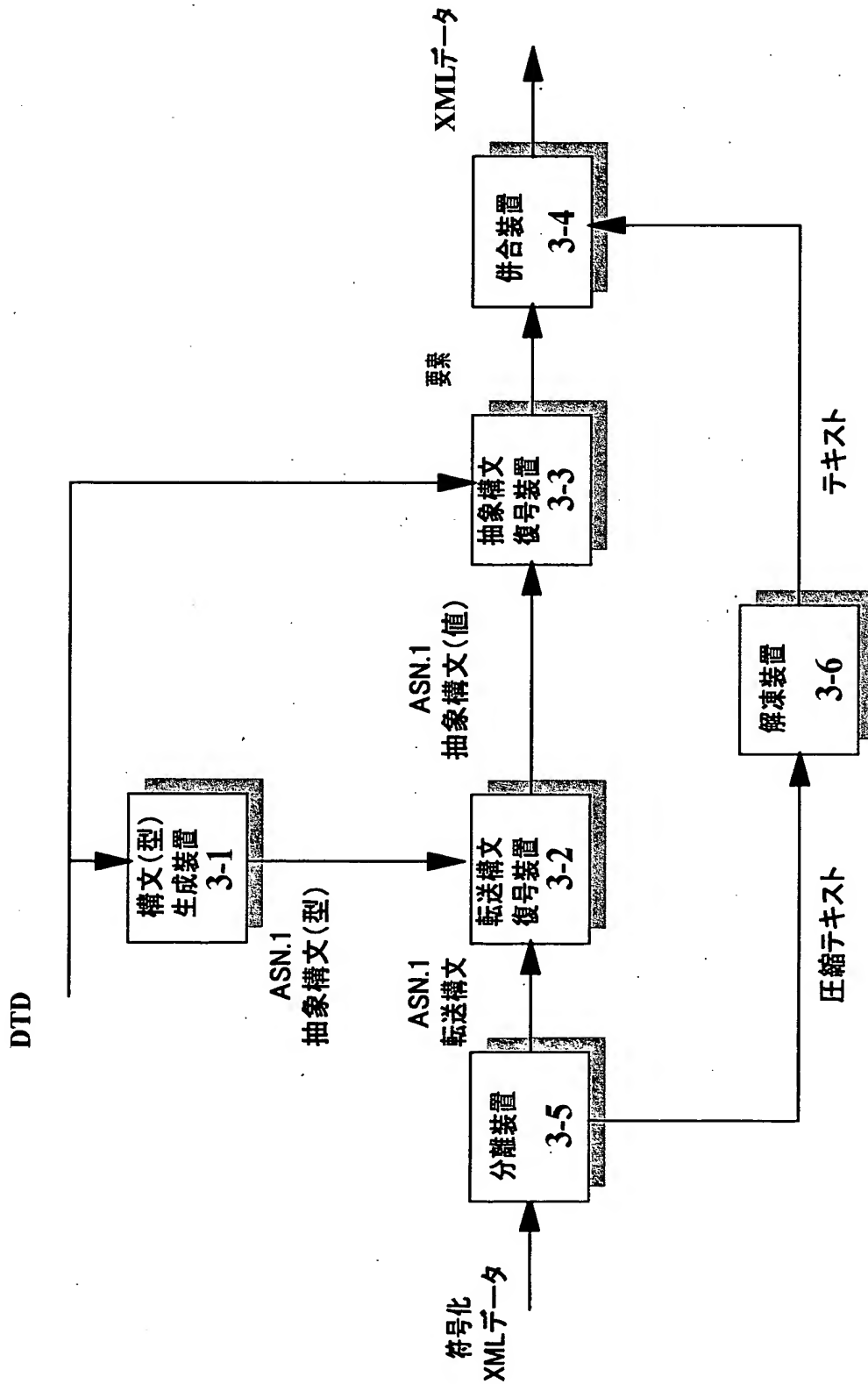
変換装置、9-5…復号化装置、9-6…XMLデータ変換装置。

【書類名】 図面

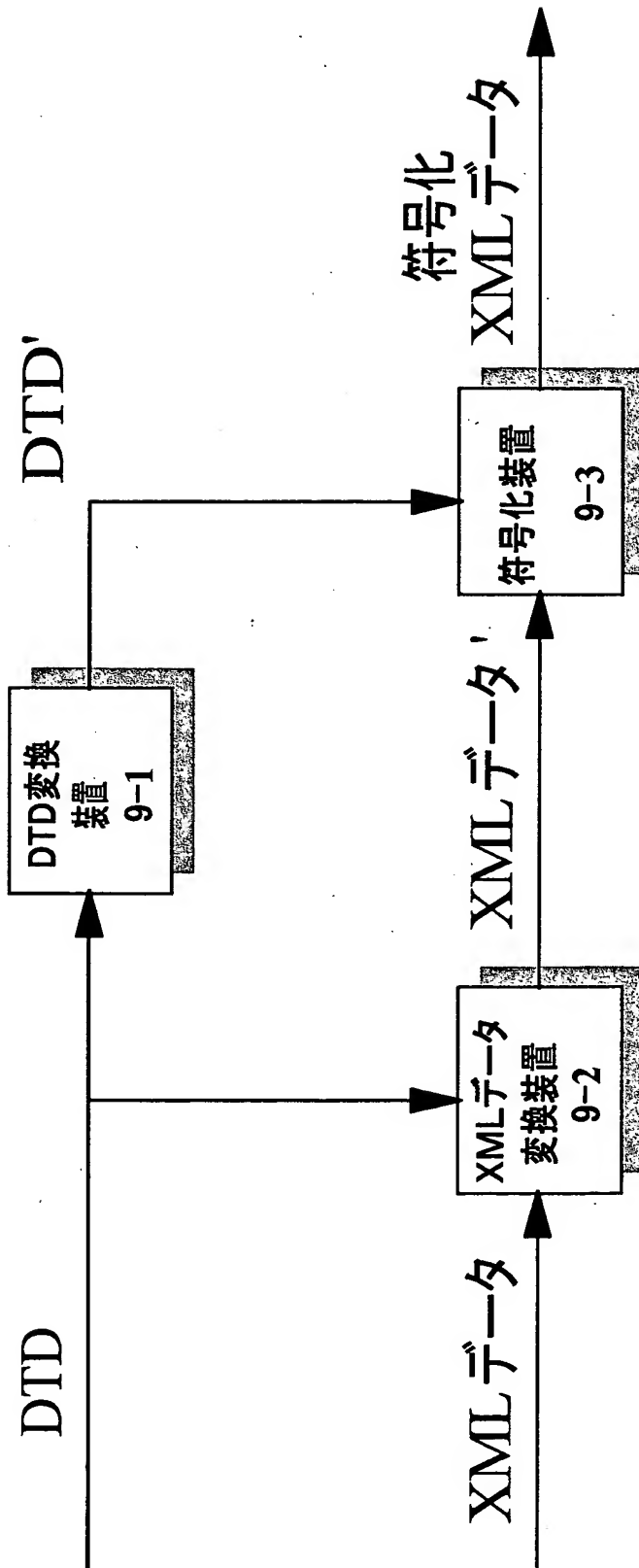
【図 1】



【図 2】

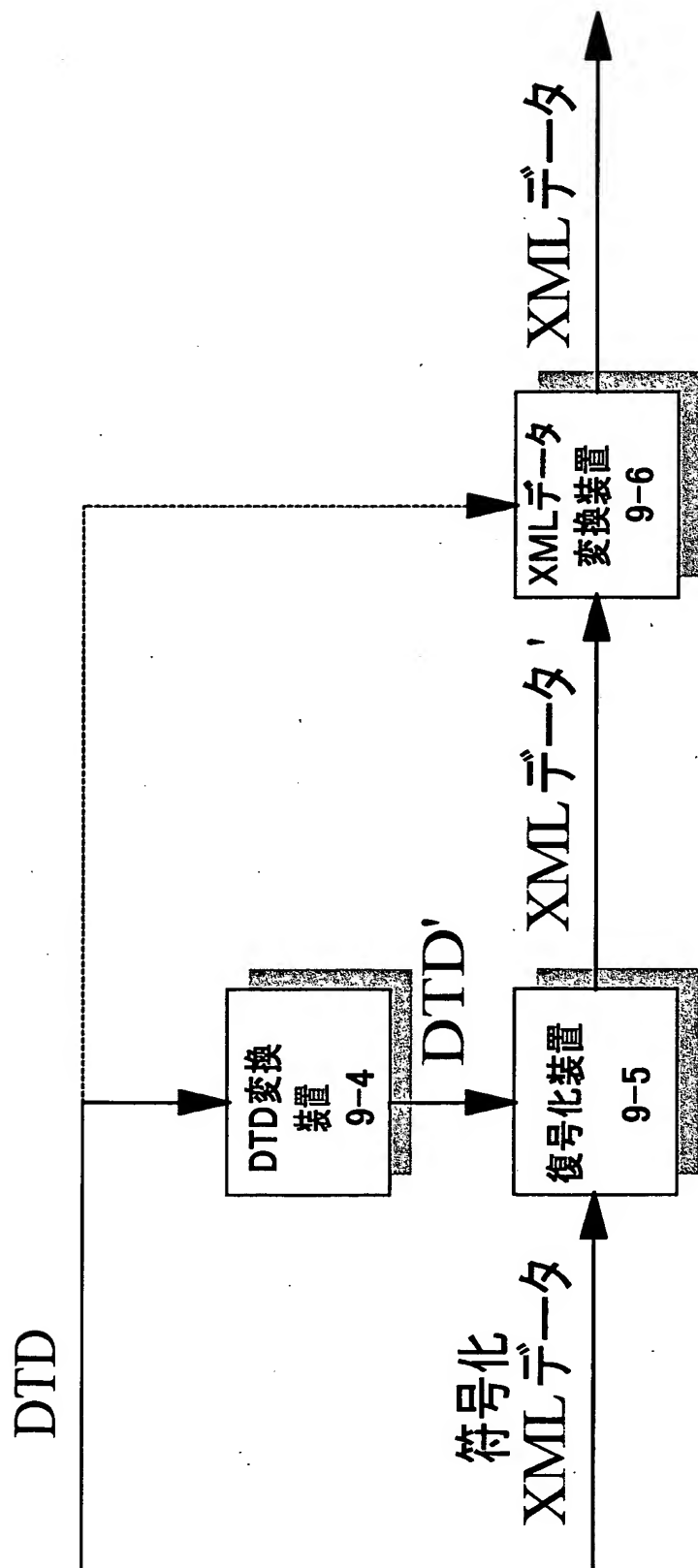


【図 3】

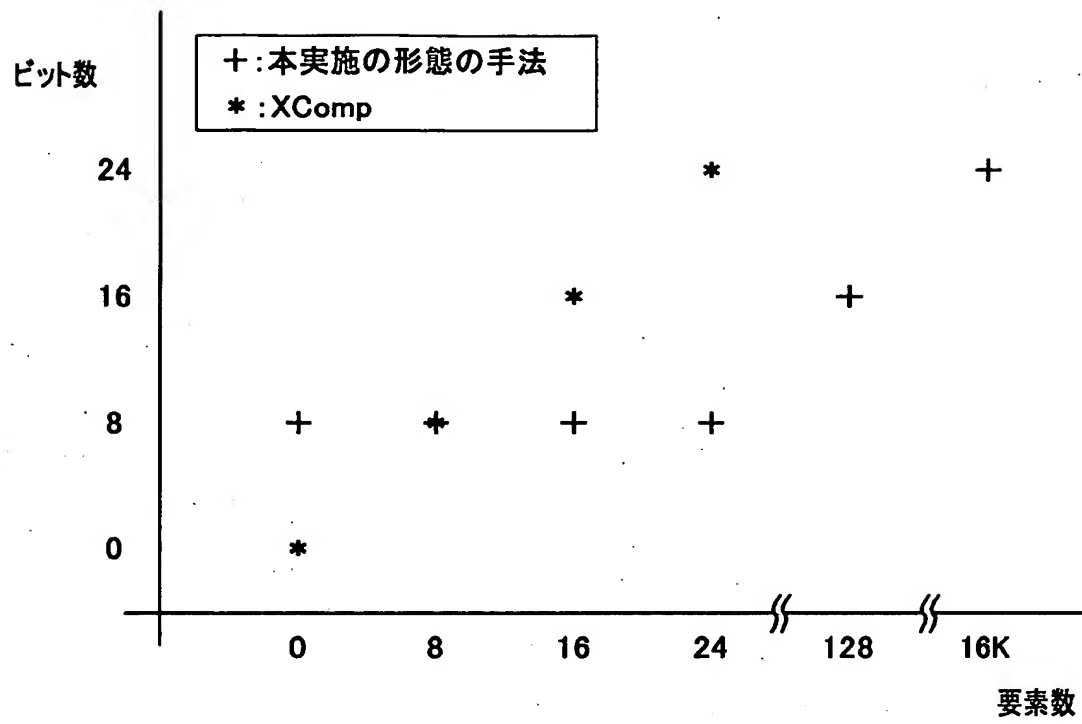




【図 4】



【図5】



【書類名】 要約書

【要約】

【課題】 XMLデータの符号化（圧縮）効率を高める。

【解決手段】 構文（型）生成装置 2-1 を用いて DTD を ASN. 1 抽象構文（型）に変換し、分離装置 2-2 を用いて前記 DTD に従う XML データからテキストを分離する。その後、構文（値）生成装置 2-3 を用いてテキストが分離された後の XML データ（要素の構文）を、前記 ASN. 1 抽象構文（型）に従う ASN. 1 抽象構文（値）に変換する。その後、転送構文生成装置 2-4 を用いて ASN. 1 抽象構文（値）を ASN. 1 転送構文に変換する。一方、圧縮装置 2-5 を用いて分離されたテキストを圧縮する。さらに、併合装置 2-6 を用いて ASN. 1 転送構文と圧縮テキストを併合し符号化 XML データを生成する。

【選択図】 図 1

認定・付加情報

特許出願の番号	特願2001-027462
受付番号	50100152058
書類名	特許願
担当官	末武 実 1912
作成日	平成13年 3月21日

<認定情報・付加情報>

【特許出願人】

【識別番号】	390009531
【住所又は居所】	アメリカ合衆国10504、ニューヨーク州 アーモンク (番地なし)
【氏名又は名称】	インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】

【識別番号】	100086243
【住所又は居所】	神奈川県大和市下鶴間1623番地14 日本アイ・ビー・エム株式会社 大和事業所内
【氏名又は名称】	坂口 博

【代理人】

【識別番号】	100091568
【住所又は居所】	神奈川県大和市下鶴間1623番地14 日本アイ・ビー・エム株式会社 大和事業所内
【氏名又は名称】	市位 嘉宏

【代理人】

【識別番号】	100106699
【住所又は居所】	神奈川県大和市下鶴間1623番14 日本アイ・ビー・エム株式会社大和事業所内
【氏名又は名称】	渡部 弘道

【復代理人】

【識別番号】	100112520
【住所又は居所】	神奈川県大和市中心林間3丁目4番4号 サクライビル4階 間山・林合同技術特許事務所
【氏名又は名称】	林 茂則

【選任した復代理人】

【識別番号】	100110607
--------	-----------

次頁有

認定・付加情報（続き）

【住所又は居所】 神奈川県大和市中央林間3丁目4番4号 サクラ  
イビル4階 間山・林合同技術特許事務所  
【氏名又は名称】 間山 進也

出 願 人 履 歴 情 報

識別番号 [390009531]

1. 変更年月日 2000年 5月16日

[変更理由] 名称変更

住 所 アメリカ合衆国10504、ニューヨーク州 アーモンク (番地なし)

氏 名 インターナショナル・ビジネス・マシーンズ・コーポレーション